caGrid 1.0: An Enterprise Grid Infrastructure for Biomedical Research

Scott Oster[1], MS, Stephen Langella[1], MS, Shannon Hastings[1], MS, David Ervin[1],
Ravi Madduri[2], MS, Joshua Phillips[4], Tahsin Kurc[1], PhD,
Frank Siebenlist[2], PhD, Peter Covitz[3], PhD, Krishnakant Shanbhag[3], MS,
Ian Foster[2], PhD, and Joel Saltz[1], PhD

[1]Department of Biomedical Informatics,
The Ohio State University, Columbus, OH

[2]Mathematics and Computer Science Division,
Argonne National Laboratory, Argonne, IL

[3]National Cancer Institute Center for Bioinformatics, Rockville, MD

[4]SemanticBits, Reston, VA

Contact Author: Tahsin Kurc
Address: Biomedical Informatics Department
Ohio State University
3184 Graves Hall, 333 West 10th Ave.
Columbus, OH, 43210
E-mail: kurc@bmi.osu.edu
Phone: 614-292-6568
Fax: 614-688-6600

Abstract

Biomedical research efforts require integrated access and management of increasingly diverse collections of information sources, analysis methods, and applications. Research efforts also frequently need to discover, characterize and access new sources of information. As a step towards supporting the above integration, access, and discovery goals, the caGrid infrastructure has been developed. It is the core Grid architecture of the NCI-sponsored cancer Biomedical Informatics Grid (caBIG^{TM}) program. The current release, caGrid version 1.0 (caGrid 1.0), is developed as the production Grid software infrastructure of caBIG^{TM}. This paper presents an overview of caGrid 1.0 and its main components.

1 Introduction

caGrid is the service-based Grid software infrastructure of the National Cancer Institute (NCI) cancer Biomedical Informatics Grid (caBIG^{TM}) program. The caBIG™ program was established to implement the enabling informatics technologies so that researchers can more efficiently share, discover, integrate, and process disparate clinical and research data resources with the overarching goal of accelerating cancer research. A wide range of development and deployment efforts, driven by the caBIG community, are underway in the program. These efforts include development of 1) a common set of applications for data management and analysis, 2) guidelines, informatics standards, and tools to support development of interoperable data and analytical resources, 3) guidelines and processes for secure sharing of data and tools, and 4) a Grid architecture that will link applications and resources in the caBIG environment. More information on caBIG™ can be obtained from the program web site: https://cabig.nci.nih.gov/.

The caGrid infrastructure is needed because basic, clinical, and translational cancer research increasingly requires integrated query and analysis of many related types of data. Moreover, there is an increasing number of cooperative groups and large scale collaborative projects, such as the Integrative Cancer Biology Program (ICBP), that collect and store data at multiple institutions. A major obstacle to more effective use of information in multi-institutional settings is the lack of interoperability among resources and efficient mechanisms to discover and securely access distributed information. Interoperability is an especially challenging problem in the biomedical research domain, because there are a variety of representations and semantics associated with biomedical data; datasets are often stored in different formats; information is represented using different terminologies; and data analysis applications are invoked and executed in different ways. The caGrid infrastructure is designed to facilitate interoperability between disparate and geographically distributed data and analytical resources, support secure electronic data interchange, and enable information discovery in a distributed environment.

This paper describes caGrid version 1.0 (caGrid 1.0), which was released to caBIG™ participants and the research community at large in December 2006. An earlier prototype version of caGrid (caGrid 0.5) is described in our previous paper (1). caGrid 0.5 was a test infrastructure used by the caBIG™ community to provide feedback on the caGrid architectural model based on a set of reference application implementations. As a result of this feedback, many changes and improvements have been made to the caGrid architecture. caGrid 1.0 is the production Grid environment in caBIG™. It provides a set of core services, toolkits for the development and deployment of new community provided services, and application programming interfaces (APIs) for building client applications. caGrid 1.0 supports: 1) integration of information from multiple data sources, 2) analyses able to leverage distributed analytic resources, 3) discovery of data sources and analytic services, 4) discovery of metadata such as concept and biomedical model definitions, and 5) support for distributed authentication and group-based security.

The architecture of caGrid and many of its core components are generic enough to be applicable in other biomedical research domains and in other distributed computing environments. In fact, several caGrid components have been contributed back for incorporation into the Globus Toolkit (2) and are seeing adoption by other communities. For instance, the Johns Hopkins led Cardiovascular Research Grid effort uses caGrid components along with components from the Biomedical Informatics Research Network (BIRN) project (3, 4). caGrid is built on open source components and caGrid source code is publicly and freely available under a liberal open source license that encourages adoption and use in both academia and industry . Additional information and downloads of the software can be found at the project web site (https://cabig.nci.nih.gov/workspaces/Architecture/caGrid).

The design and development of caGrid 1.0 was a community based effort led by the Ohio State team in close collaboration with the National Cancer Institute Center for Bioinformatics (NCICB) and with substantial contributions from the Argonne group. The caGrid effort leveraged the previously existing Ohio State Mobius grid infrastructure, the NCICB caCORE SDK, caDSR and EVS and the Argonne based Globus toolkit. The Ohio State Grid Authentication and Authorization with Reliably Distributed Services (GAARDS) security infrastructure and the Introduce toolkit, originally developed as part of caGrid 0.5, and Globus were substantially generalized and enhanced through the caGrid 1.0 community design process.

2 Objectives and Overview of caGrid

The caGrid infrastructure is designed to support a wide range of use cases in basic, translational, and clinical cancer research. For purposes of description, we group these use cases into three overlapping categories: 1) discovery, 2) integrated and large-scale data analysis, and 3) coordinated study. The first category of use case involves targeted searches that return precisely defined attributes and data values from heterogeneous information sources. The second use case category involves the access, aggregation and integrated analysis of multiple types of data, including clinical, molecular, image, pathology, and cytogenetic data. The third use case category is motivated by requirements posed by cooperative groups and multi-institutional clinical trials. This last use case category involves coordinated accrual, access, and analysis of information by multiple remote researchers and across multiple institutions.

In order to support these use cases, caGrid is built as a Grid software infrastructure and leverages Grid computing technologies and the Web Services Resource Framework (WSRF) (5) standards. Grid computing(6, 7) was initially conceived to enable secure access to high-end computing facilities across multiple supercomputer centers. Technologies and standards developed by the Grid computing community (http://www.ogf.org) have evolved it into a platform to facilitate development, sharing, and integration of distributed data and analytical resources and to support applications that make use of those resources. Grid computing has been employed in a wide range of projects and applications in biomedical research (3, 4, 8-18). A common theme across these different application projects is the need for support for query, retrieval, and integration of data from heterogeneous resources, discovery of relevant resources, and analysis of data using one or more analytical services. The caGrid 1.0 infrastructure aims to support such biomedical research applications by creating a Grid infrastructure wherein the structure and semantics of data can be determined programmatically and distributed resources can be discovered and accessed from within application programs. caGrid has several unique features that differentiate it from other Grid computing projects. These features, which will be described in more detail in Section 4, can be summarized as follows:

- caGrid is built on a Model Driven Architecture (MDA) best practice with emphasis on syntactic and semantic interoperability across heterogeneous resources. caGrid's client and service APIs present an object-oriented view of data and analytical resources, and operate on data types based on common data elements and controlled vocabularies.
- It facilitates metadata-driven discovery of resources by taking advantage of structural and semantic descriptions of data models and services. In caGrid 1.0, a client can discover resources based on associated data models and semantic information.
- It provides a comprehensive security system, the GAARDS infrastructure. GAARDS leverages Grid security infrastructure mechanisms (19, 20) to provide Grid-wide identity management support, support for management of Grid-wide trust fabric, and support for virtual organizations and Grid-wide group management.

- It provides a unified service authoring toolkit, called Introduce (21), for service providers to easily develop and deploy caGrid compliant data and analytical services.
- It implements a workflow service for orchestration of Grid services. The workflow management service uses the industry standard Business Process Execution Language (BPEL) (22), enabling the execution and monitoring of BPEL-defined workflows in a secure Grid environment.
- It enables federated access to distributed data sources via a common data service and federated query infrastructure. The federated query support enables distributed aggregations and joins over multiple data services.
- It provides support for bulk data transfer to allow uniform access to large datasets from caGrid compliant services. This work currently supports access via the existing specifications of WS-Enumeration(23), WS-Transfer(24), and GridFTP (25).

In the next section we describe an example application scenario that is based on the *discovery use case* and the *integrated and large-scale data analysis use case*. We will use it throughout the paper as our recurring example to illustrate the design and implementation of the caGrid 1.0 components and how the various features of caGrid 1.0 can be employed.

3 Example Application Scenario: Evaluation of CAD Methods for Screening

Our application scenario is a simplified version of the GridIMAGE effort (26, 27), which uses caGrid 1.0 as the underlying software infrastructure. It combines elements of the discovery and the integrated and large scale data analysis use cases. In this scenario, a researcher wants to evaluate his computer aided detection (CAD) algorithm to screen for clinically significant lung nodules. The CAD algorithm characterizes texture and three dimensional shapes and identifies lung nodules based on this characterization. The researcher wants to access multiple image repositories that contain chest CT imagery, retrieve images from these repositories, and process them with his CAD algorithm and other algorithms for comparison. The CAD algorithm developed by the researcher and other algorithms are wrapped as caGrid analytical services. Similarly, image archives are exposed as caGrid data services.

In caGrid, when a service is deployed, the service is required to publish metadata about itself. Metadata associated with a caGrid data service can specify, for example, what types of image data are served by the source and whether longitudinal follow-up imagery is available for patients. Metadata for a caGrid analytical service may include information about what analytical methods are exposed by the service, what data types each method takes as input, and what data types each returns as output. The service metadata is registered in the caGrid Index Service, which allows searches on metadata elements. The researcher uses the Index Service to discover caGrid image data services that contain chest CT imagery. He/She also searches for existing caGrid analytical services that provide CAD analysis functions for CT imagery, to which the researcher will compare his/her algorithm. Both discovery queries make use of published controlled terminology to ensure that discovered services share common data semantics.

Once the relevant services have been identified, the researcher selects subsets of images from these services based on the availability of follow-up imaging studies and the availability of outcome related correlative data. Using caGrid, he/she formulates and executes an image analysis workflow to assess his/her algorithm's value as a screening tool. This workflow 1) queries image archives to identify and access the chosen images and correlative metadata, 2) moves the chosen images to the machines running the caGrid analytic services, which implement CAD algorithms (including the researcher's own algorithm), 3) iteratively invokes the CAD algorithm at each service on each image, and 4) loads the nodule location, shape and texture results obtained by the CAD algorithms into another database (so that the results can be queried and accessed later by the researcher's collaborators); this application-specific database also can be implemented as a caGrid data service. Following completion of the workflow, the researcher can then query the results database and carry out further statistical analyses to evaluate his/her algorithm for use in screening patients.

The use of caGrid in the application scenario has several advantages. First, there is no need to collect all data to a centralized location or to host the various analysis programs on a central server. Each institution can set up their local data repositories and make them available to others in a controlled manner, via standard data exchange protocols (as provided by the caGrid infrastructure). Similarly, analytical services can be hosted at different institutions and accessed by standard communication protocols. Second, programmatic interoperability across different resources becomes possible, because caGrid services are required to operate on well-defined, registered objects and expose their functionality through well-defined interfaces (see Section 4.1). Client applications (e.g., review clients) can interact with different resources without having to implement codes specific to each backend system. Third, the use of common terminology and rich service metadata (see Section 4.2) allows for metadata driven discovery and automated invocation (via workflows) of data and analytical sources, and makes sharing of tools and resources more effective. A researcher can search for and locate resources that could satisfy a particular request (e.g., all image repositories that have CT images). Fourth, security, authentication, and authorization policies can be implemented and enforced to control access to resources in a multi-institutional environment. Each institution can protect their intellectual property, patient privacy, and resources, while also allowing for sharing of information and tools for collaborative efforts.

4 System Architecture

A primary principle of caBIG™ is the use of open standards. caGrid is built as a service-oriented architecture based on standard Web Services specifications (28)—in particular, those standards defined within the Organization for the Advancement of Structured Information Standards (OASIS) and elsewhere for representing service state (Web Services Resource Framework: WSRF; WS-Notification(5)) and security (WS-Security, Security Assertion Markup Language: SAML, Extensible Access Control Markup Language: XACML). It aims to be programming language and toolkit agnostic. Each data and analytical resource in caGrid is implemented as a Web Services Resource Framework (WSRF) v1.2 compliant Web Service, which interacts with other resources and clients using Web Services protocols. The implementation of the infrastructure makes use of several Grid systems, including the Globus Toolkit(2) and Mobius(29), and tools developed by the NCI such as the caCORE infrastructure(30). More detailed description of these systems and tools can be found in the Appendix section.

The caGrid infrastructure provides a set of core coordination services and a runtime environment to support the deployment, execution, and invocation of caGrid data and analytical services. The core services provide support for common Grid-wide operations required by clients and other services. These operations include metadata management; advertisement and discovery; federated query; workflow management; and security. Figure 1 shows the production deployment of the caGrid infrastructure with coordination services (e.g., metadata services, security services, workflow service) and community-provided data and analytical services. caGrid coordination services are designed to be replicated and distributed for purposes of reliability and scalability. Users access caGrid services via application specific client programs or web portals. A suite of client applications, including the Security Administration Graphical User Interface (GUI) and the Introduce toolkit, are available for system administrators and developers to interact with some of the core services and to develop and deploy application-specific services. caGrid 1.0 implements several important features on top of the basic Grid architecture to better address the informatics needs of cancer research. In the rest of this section we present these key features.

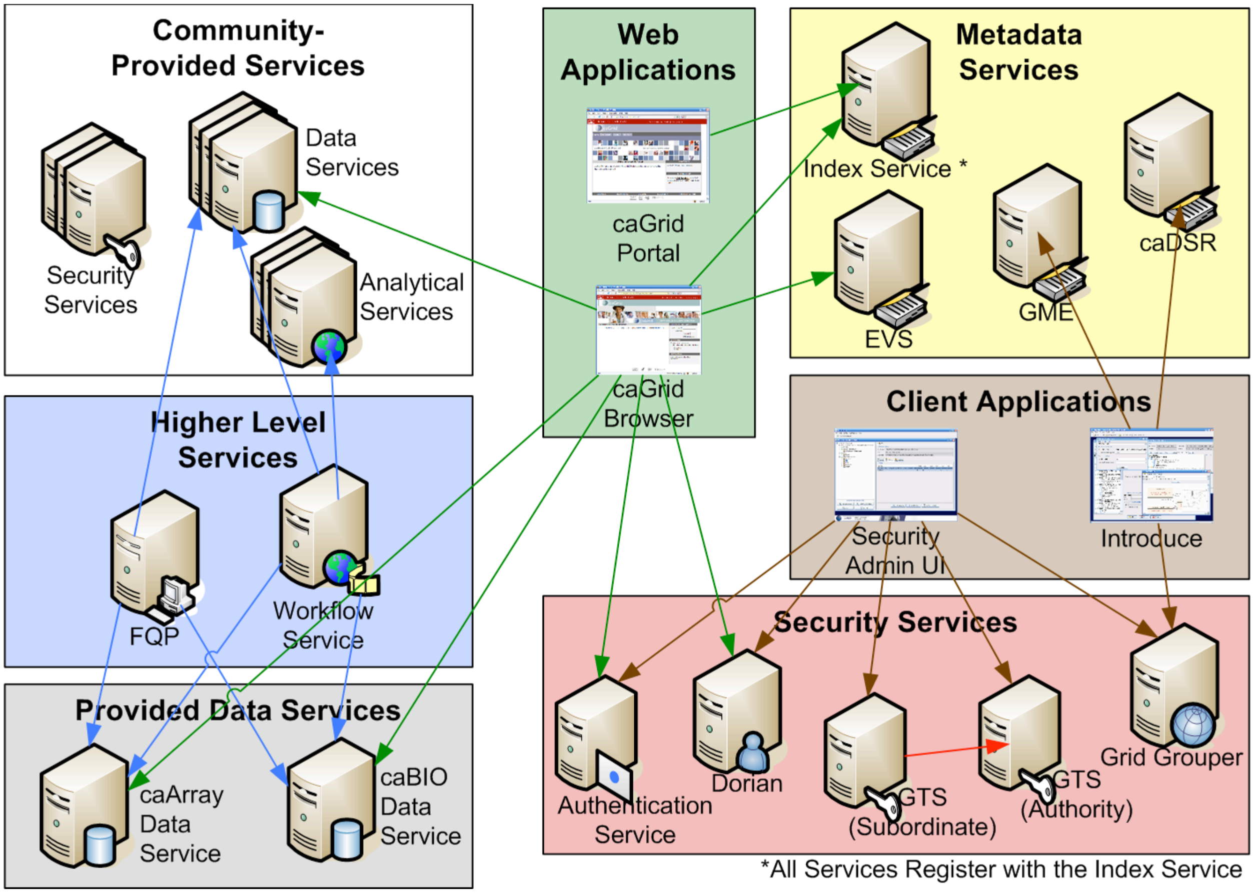

Figure 1. caGrid 1.0 infrastructure and environment. The core caGrid services include the security services (Dorian, Grid Trust Service (GTS), and Grid Grouper), metadata services (Index Service, Global Model Exchange (GME), Enterprise Vocabulary Services (EVS), and cancer Data Standards Repository (caDSR)), and high level services such as the Federated Query Processing service (FQP) and the Workflow services. In addition to the core services, data services and analytical services (e.g., caArray and caBIO services in the figure), which are provided by research groups, institutions, individual researchers, can be discovered and securely accessed using the caGrid core services and protocols. Common invocation patterns are indicated by directional arrows.

4.1 Interoperability

4.1.1 Syntactic and Semantic Interoperability

Effective sharing of data and tools in an environment is hindered greatly if resources developed and deployed by different groups of scientists are not interoperable. *Syntactic interoperability* facilitates uniform and programmatic access to otherwise heterogeneous resources. *Semantic interoperability* is needed to ensure correct interpretation of information served by a resource. To enable syntactic and semantic interoperability among data sources and applications, the caBIG^{TM} community has developed guidelines and a set of requirements to represent the interoperability level of an application in terms of vocabularies, data elements, domain models, and APIs. These levels (Legacy, Bronze, Silver, and Gold) and guidelines are outlined in the caBIG^{TM} compatibility guidelines document(31). Resources that conform to the requirements of the Silver level are considered syntactically and semantically interoperable. Silver level indicates that a resource has well defined APIs that provide object-oriented access to backend resources. Information hosted by a backend resource is expressed in registered domain models, which define object classes and associations among them. An object class is made up of common data elements, which are semantically annotated by community curated, controlled vocabularies. Silver level domain models are defined in Uniform Modeling Language (UML) and converted into common data elements. The common data elements are managed in the cancer Data Standards Repository (caDSR)(30). The definitions of these data elements draw from vocabulary registered in the Enterprise Vocabulary Services (EVS)(30). In this way, the concepts of data elements and the relationships among them are semantically annotated. Semantic model annotation includes the meanings of individual data elements, their value domains (the set of values an attribute is permitted or not permitted to store), and the concepts associated with data elements.

The Gold level requirements extend the Silver level requirements to also include 1) WSRF service interfaces and registered XML schemas for data exchange and 2) a common framework across the caBIG federation for the representation, advertisement, discovery, security, and invocation of distributed data and analytic resources. caGrid provides the tooling for developers to produce a Gold level application. In caGrid, XML schemas corresponding to common data elements and object classes are registered in the Mobius GME service. *In sum, the caDSR and EVS define the properties and semantics of caBIG™ data types, and the GME defines the syntax of their XML materialization.*

Note however that use of caGrid infrastructure does not obligate a group to adopt the caBIG^{TM} silver level semantic compatibility standards and process; other communities such as the Cardiovascular Research Grid (described in Section 1) use caGrid components but employ different criteria and standards for interoperability. Moreover, while caGrid aims to enable interoperability among disparate data and analytical resources, it does not enforce that all resources represent information in an all-encompassing, single domain model, or a meta-schema. The infrastructure provides functionality for clients to discover, examine, and use domain models and XML schemas in order to interact with distributed and heterogeneous services.

4.1.2 Interaction with Resources and Communication

Client and service APIs in caGrid operate on instances of objects that conform to domain models registered and published in the environment. Communication between the client program and the analytical and data services is carried out using standard messaging protocols (i.e., WSRF protocols). Messages exchanged between two caGrid end-points and message payloads (i.e., the data transferred in messages) are encoded in XML. When an object is transferred between clients and services, it is serialized into a XML document that adheres to a

XML schema registered in the Mobius GME. With a published model and schema, the receiving end-point can parse the data structure and interpret the information correctly.

4.1.3 Example Application Scenario

In the application scenario described in Section 2, a developer (or a group of developers) in could create an object model for CT chest image data and an object model for the CAD analysis results. The model for CT chest images might, for instance, contain data elements to store imaging instrument parameters, information about the study, to which the image data belongs, as well as the image data itself. This model would typically be developed using a UML modeling software with vocabulary and meanings drawn from the EVS. The caBIG compatibility guidelines place emphasis on reuse in developing domain models. That is, domain models developed in different projects should reuse existing vocabulary terms and common data elements, when appropriate and possible, for better interoperability. For example, if there were common data elements in caDSR that encode information about CT image markup and annotation, the CT chest data model in our example could reuse these data elements to better support sharing and dissemination of research study data[1]. The caBIG^{TM} project is organized to facilitate vocabulary and common data element reuse through a systematic peer review process. Once a model is finalized through this review process, the objects in the model are registered in the caDSR and any necessary additions are made to the EVS. Following the creation and registration of the CT chest image model, an XML schema for the model is created and stored in the Mobius GME. This schema describes the syntactic representation of data from the model, as it is passed on the Grid, and is the basis for the development grid service interfaces which use data types from the model as parameters of their operations.

[1] The caBIG In-vivo Imaging Domain Workspace is developing common data elements and terminology for annotation and image markup. With the availability of such community accepted data elements and vocabulary, the developers in our example could choose to reuse those data elements for better interoperability with other imaging applications in the in-vivo imaging domain.

In our example scenario, analysis programs are wrapped as caGrid analytical services and image archives as caGrid data services. The owners of image archives expose their CT chest data through the CT chest data model and the CAD algorithm services can implement an interface that will take objects of this model and operate on them. In this way, client programs can safely parse and interpret the data retrieved from services and can submit requests that can be processed correctly by the services. The functionality of the backend CAD analysis program is exposed through a set of service-side and client-side interfaces that can be invoked by remote client programs. Using the client-side interfaces, a client can send a request to the CAD analysis service to process a set of images, or submit a query to the caGrid image data service to find CT chest images from a particular study and retrieve them. The client program and the services communicate with each other via standard WSRF messaging protocols. While not a unique feature to caGrid, a novel aspect of this approach is that idiosyncrasies of backend systems and platform or language specific restrictions, which are prevalent in this domain, are completed isolated from the client program through the service oriented use of standardized protocols.

4.2 Metadata Support and Semantic Discovery of Resources

Each caGrid service is required to describe itself using standard, yet extensible, service metadata. At the base is the common service metadata, to which every service is required to adhere. This metadata contains information about the service-providing cancer center, such as the point of contact for the service and the institution’s name which is providing the service. It also describes the objects used as input and output of the service’s operations. The definitions of the objects themselves are described in terms of their underlying concepts, attributes (data elements), and associations to other objects as extracted from the caDSR and the EVS. Referencing these object definitions, the common service metadata specifies the operations or methods the service provides, and allows semantics to be applied to them. The base common metadata is extended for different types of services. Data Services, for example, provide an additional “domain model” metadata standard. This metadata details the domain model, including associations and inheritance information, from which the objects being exposed by the service are drawn. In the example application scenario, the metadata for each image data service would specify information about the cancer center hosting the data service, the names of the contact points, which may be the owner of the datasets, and the CT chest object model as well as object models for other image types maintained by the data service. For analytical services, the metadata would include the definitions of the object models that are input to and output from each analytical operation, as well as standard metadata elements.

When a service is deployed in caGrid, its service metadata is registered with an indexing registry service, called the Index Service, provided by the Globus Toolkit. The Index Service can be thought of the repository of information about all advertised and available services in the environment. A researcher can discover services of interest by looking them up in this registry. caGrid 1.0 provides high-level APIs for performing searches on these metadata standards, thus facilitating discovery of resources based on data models and semantic information associated with them. Using the caGrid 1.0 discovery APIs, our researcher can, for instance, search for and locate all services from a given cancer center, data services exposing CT chest object models, and analytical services that provide operations that take CT chest data and return annotated image information. The caGrid 1.0 metadata infrastructure, APIs, and toolkits are defined with extensibility in mind, allowing domain or application specific extensions to the advertisement and discovery process.

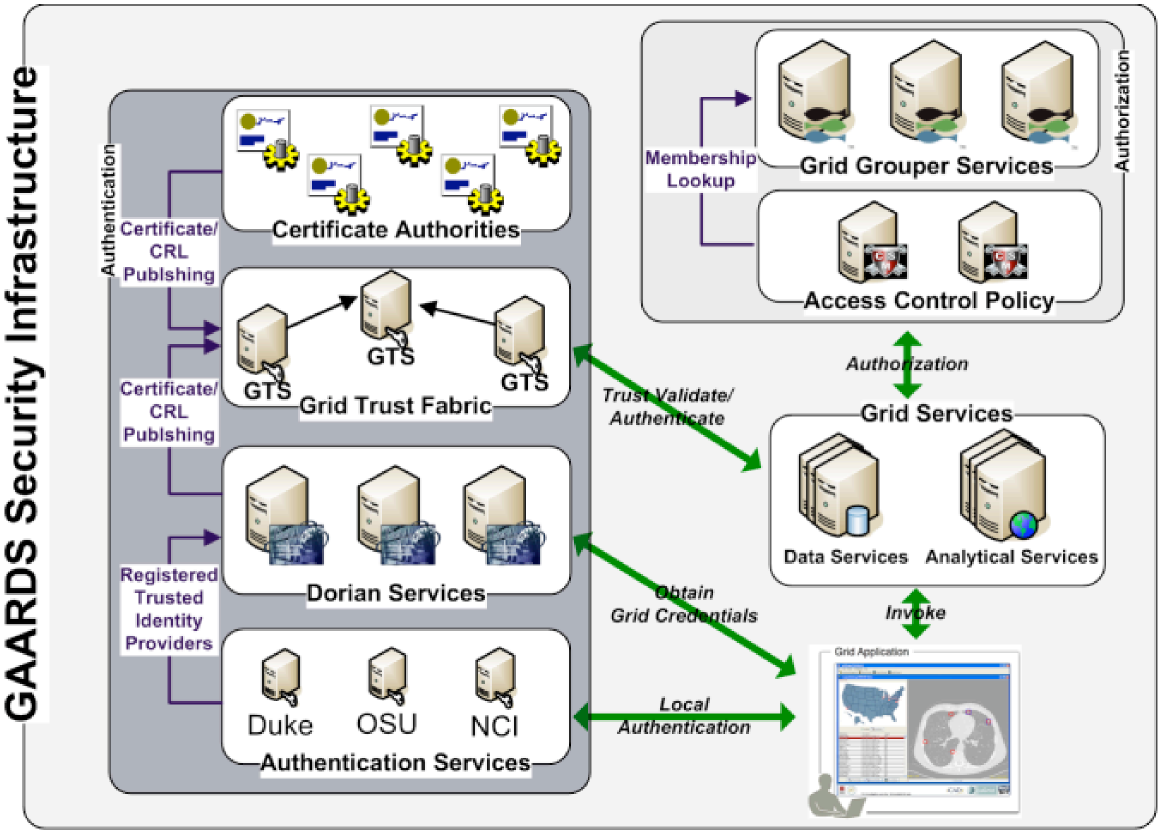

Figure 2. The GAARDS security infrastructure and its deployment in a multi-institutional environment. It consists of Dorian for federation and management of grid credentials, the Grid Trust Services for management of a trust fabric in the environment, and Grid Grouper for facilitating authorization and access control. When a Grid client wants to access a secure caGrid data or analytical service, the client can use his/her local authentication mechanisms and Dorian to get a temporary grid credential (based on the long term certificate and private key of the client, which are managed by Dorian). The client can then interact with the secure caGrid service and invoke its methods with his/her grid credentials. The caGrid service may contact the Grid Trust Services infrastructure to check if the client's credentials are valid (i.e., they are not revoked; Certificate Revocation Lists: CRLs are used to keep track of revoked certificates) and if the credentials are issued by a trusted identity provider. If the authentication of the client is successful, the caGrid service may contact the Grid Grouper service to obtain the client's group information upon which access control policies may be defined. Depending on the authorization information, the caGrid service may deny the client's access to some or all service methods or the data served by the service.

4.3 Security

Translational research projects have crucial security requirements needed for patient privacy, to enforce data access requirements stipulated by institutional review boards (IRBs), and to protect intellectual property. The Grid Authentication and Authorization with Reliably Distributed Services (GAARDS) infrastructure of caGrid 1.0 provides services and tools for the administration and enforcement of security policies in a multi-institutional environment. It implements a mechanism to create and manage virtual, federated organizations among collaborating institutions. GAARDS extends existing Grid based security mechanisms (32-34). It consists of three main components: Dorian for provisioning and federation of Grid user identities and credentials(35); Grid Trust Service (GTS) for maintaining a federated trust fabric of all the trusted credential providers in the Grid (36); and Grid Grouper for group-based authorization support. The GAARDS infrastructure is illustrated in Figure 2. In the following sections, we describe individual components of the GAARDS infrastructure. We illustrate how GAARDS can be employed using our example scenario in Section 4.3.4.

4.3.1 Dorian: Provisioning of User Accounts

A key security requirement is to be able authenticate users so that access to resources can be restricted to individual users or a group of users. A user in the Grid is identified by an X.509 Identity Certificate. The X.509 Certificate with its corresponding private key forms a unique credential (“grid credential”) for the user. The GAARDS Dorian service provides the support 1) for system administrators to manage and federate user identities in the Grid and 2) for users to obtain Grid credentials by using organization-provided credentials. Dorian is designed to address several problems in large, distributed environments such as caBIG^{TM}, whose community is growing and envisioned to span hundreds of institutions and thousands of researchers. Manual creation and management of user Grid credentials in such environments is error prone and not scalable because of the large number of users and user organizations. If users want to authenticate from multiple locations, a copy of the users' certificates and private keys need to be replicated at each of those locations. This poses a security risk because securely distributing certificates and private keys is a difficult process. Moreover, organizations have their own policies and processes in place for creating user accounts and systems for managing these accounts. By leveraging existing systems, it is possible to implement a scalable, efficient system for assigning user Grid credentials. This approach allows institutions to enforce their own policies for management of local user accounts.

Dorian serves as an integration point between institutional security domains and the caGrid environment for federation of user identities; Dorian hides the complexities of managing Grid credentials and issuing proxy certificates to users. A user can use his/her institutional account to get Grid credentials. Dorian makes use of the SAML assertion framework (37) as the underlying technology for federation of user identities and authentication of a user to obtain a Grid credential as a proxy certificate. Dorian maintains the long term certificate and private key for its users, and allows users access to short term proxy certificates (signed by each user's long term credentials) in exchange for corresponding SAML assertions. It will only issue grid credentials to users that supply a SAML assertion signed by a trusted Identity Provider, which meets the administrator configured authentication policies. In order for users to use their existing accounts, the organization providing these accounts must be registered with Dorian as a trusted Identity Provider. Dorian also provides an internal user management and accounting system which can be used as a standalone Identity Provider for unaffiliated users or when integration with institutional Identity Providers is not required.

4.3.2 Grid Trust Services: Management of Trust Fabric

Trust is an important aspect in the authentication and authorization process. Institutions will have different policies for granting credentials and enforcing these policies. They will employ different security systems for authenticating users to the environment. It is thus conceivable that there will be different levels of trust between service providers and service consumers. When a client contacts a secure caGrid service, the service should check if the user's credentials are issued by a trusted certificate authority and whether the user's credentials have been revoked or not. Similarly, the client should make the same checks on the service to verify it should trust the service to which it has connected. The service and client need to maintain a list of trusted certificate authorities using the most recent trust information and implement mechanisms to check credentials against certificate revocation lists (CRLs). This is a complicated problem in an environment where there may be hundreds of certificate authorities and thousands of credentials issued and revoked by these authorities and where new certificate authorities are added to the environment dynamically. The Grid Trust Service (GTS) component of GAARDS provides a Grid-wide solution to the problem of maintaining trusted certificate authorities, managing trust relationships, and enabling users and services to make authentication and authorization decisions using the most recent trust information (36). In sum, a service developer can use the GTS to manage the list of trusted certificate authorities and CRLs based on trust levels and the most recent trust agreements for a given service. The GTS enables the definition and management of levels of assurance, or trust, so that certificate authorities can be grouped, managed, and discovered by the trust level acceptable to the service. Similar to services' usage of the GTS, client applications can use the GTS to find services whose certificates are issued by authorities that meet the trust level requirements of the client; a client can submit a certificate and trust requirements in exchange for a validation decision. The GTS is a federated system in that it allows for autonomous management of local GTS instances, yet presents the distributed

GTS instances and the whole trust fabric as if it is implemented in a centralized system. For example, caBIG^{TM} is adopting a tiered classification of certificate authorities, based on the e-Authentication standards as specified by the National Institute of Standards and Technology (NIST) Special Publication 800-63 Version 1.0.2 entitled *Electronic Authentication Guideline*. GTS will be used to disseminate this trust fabric to all caBIG^{TM} participants, but by adding local specialized GTS instances as subordinates to the larger trust fabric, institutions or collaborating parties may extend their local environment by adding internal certificate authorities which are not otherwise part of the larger trust fabric.

4.3.3 Grid Grouper: Authorization/Access Control

It is important that a service provider be able to implement access control policies locally, and at the same time enforce them based on Grid-level information (e.g., the Grid-level privileges of a client). Most access control systems and policies are implemented to grant access permissions for groups and users are given privileges based on group memberships. In the caGrid environment, the security infrastructure should assist the creation and management of groups spanning organization boundaries. The Grid Grouper service provides this infrastructure, which builds on the Grouper framework from the Internet2 initiative (http://middleware.internet2.edu/dir/groups/grouper/). It implements a service interface to the group model of the Grouper framework. Inheriting from the Grouper, it supports subgroups, composite groups (as the union, intersection, or relative complement of two other groups), custom group types and custom attributes, and basic group management by distributed authorities. A local authorization system can base its access control policy on group membership expressions over the groups managed by Grid Grouper as well as any local groups or policies it maintains. The Common Security Module (CSM) of the caCORE infrastructure is an example of such a local system. CSM provides a mechanism for managing and enforcing access control policy locally. It has been extended to enforce access control based on groups registered in Grid Grouper and its local groups. CSM is used in the GAARDS infrastructure as the default access control system that can be deployed at a site. A caGrid service asks the local CSM for authorization decisions (e.g., whether a client can perform a certain operation or not) based on the access control policy maintained in CSM. CSM reaches an authorization decision by asking Grid Grouper whether the client in question is a member of the groups specified in the policy as well as by checking the client's membership to its local groups.

4.3.4 The Use of GAARDS in Example Application Scenario

The GAARDS components can be employed in the example application as follows. We will consider two scenarios to illustrate the use of Dorian. In the first scenario, the researcher is affiliated with Institution A, which is registered with an instance of the Dorian service as a trusted identity provider. The instance of the Dorian service may be an instance managed locally by Institution A or a centralized one (e.g., an instance managed by the NCI). To obtain a proxy certificate, the researcher first logs on to a system in his/her institution through the institution's security/authentication mechanisms (e.g., the researcher logs on to his/her institutional account using her workstation). The institution's security infrastructure issues a SAML assertion indicating that the researcher has successfully authenticated. This SAML assertion is sent to Dorian automatically by the institution security infrastructure, if configured so, or manually by the user. Dorian will then issue a proxy certificate for the researcher providing the SAML assertion meets the authentication policies associated with the user's identity provider. In the second scenario, the researcher's institution may not be part of the caGrid security environment (i.e., it is not registered as a trusted identity provider) or the researcher may not want to use institution-provided credentials for authentication. In that case, the researcher can request an account, with Dorian's internal identity provider, through the caGrid portal or security application this could be a request for an account managed by a Dorian instance supported by the NCI, for example. Once the researcher has obtained approval for the account, he/she can log on to the environment using his/her application of choice which supports Dorian-based authentication. Just as in the first scenario, Dorian will issue a proxy certificate for the researcher that can be used for authentication. Once the researcher has obtained a proxy certificate, he/she can interact with secure services and can be authenticated by the services. caGrid provides standardized tooling for integrating such authentication scenarios into applications, so the general user experience for these scenarios is just a familiar "login screen."

The researcher is also assigned to one or more groups managed by a Grid Grouper instance, possibly managed by the institutions collaborating in the same project in which the researcher works. Assume these groups are "researcher", "researcher from Institution A", "CAD developer", and "lung cancer investigator". Developers of the services for image archives and CAD algorithms can implement access control mechanisms based on membership expressions over these groups as well as other groups or policies managed locally. An image data service may enforce rules that only users, who are "lung cancer investigators" and are from "Institution A", can access their chest CT data. Another image data service may only allow access to its data by individuals who are in the "physicians" group. With GAARDS in place, when the researcher in our example accesses these services, he/she is first authenticated using his/her grid credentials. If the authentication step passes with success, the researcher will be granted access to the image data service which allows "lung cancer investigators", but not to the other service, which requires the client be a physician. Similarly, the researcher can be confident that sensitive data sent to data services is protected, as he/she can validate the identity of services with which he/she communicates. If there is ever a known security breach in one of the services the researcher uses, the GTS infrastructure will ensure this service is no longer in the trust fabric and the clients will refuse to communicate with the service. A security breach may occur if, for example, the Grid certificate of the service is compromised. In this case, the attacker may try to set up a fake service using the same certificate. Since the GTS infrastructure deletes the certificate of the service from the trust fabric, a client accessing the fake service can find out that the certificate (hence the service) should not be trusted.

4.4 Support for Service Development and Deployment

caGrid is intended to be used by service developers with different levels of experience with Grid computing, Grid technologies, and implementation of Grid services. We have observed that existing Grid toolkits and middleware systems require that a service developer have good knowledge of Grid computing and low-level details of Grid toolkits (e.g., which command-line tools to call, in what order, the directory structure of service and configuration files, and the low-level details of the security infrastructure). This requirement presents a major barrier to the adoption of the caGrid infrastructure in cancer and other biomedical research domains. Moreover, caGrid services are required to be *strongly-typed*. That is, the methods of the service should consume and produce data elements that are well-defined and published in the environment. This requirement is not addressed by the existing Grid application development tools. The Introduce toolkit(21) has been developed as an extensible Grid service authoring tool with a graphical workbench for developers to easily develop and deploy strongly-typed services. Introduce reduces the development time and knowledge required to implement and deploy Grid Services by abstracting away the details of the WSRF specification and integration with the existing low-level tools. It provides a graphical user interface (GUI) and high-level usage-oriented functions and manages all the service-specific files and directories required for correct compilation and deployment. It generates appropriate, object-oriented client APIs that can be used by client applications to interact with the service. Furthermore, Introduce is extensible via plug-ins, which allow for extensions to support common service types in an application domain and to implement mechanisms for customized discovery of common data types in creating strongly-typed services. It integrates with the GAARDS infrastructure to enable development and deployment of secure services. In this way, service developers can define in a graphical manner, the authentication and authorization policies for their services at the service level as well as at the level of individual methods or resources provided by the service.

A developer in our example application scenario can use the Introduce GUI to create a basic service skeleton for CAD analytical services. Once the basic service skeleton has been created (with all files and directories required for compilation and deployment), the developer can add other service methods using the tool, e.g., a *CAD_ProcessData* method for the CAD analytical service. For each service method, he/she can choose the input and output object types of the method from a list of object models and data elements available through type management services such as GME or caDSR. He/she might then choose the CT chest image object model as the input data type and the CAD analysis result model as the return data type for the *CAD_ProcessData* method. Introduce generates the client side API to interact with the service and this method. It also creates all the code needed to convert objects to and from XML documents, which conform to the respective XML schemas in GME, for transmission across the Grid. The developer may choose to implement authentication and authorization through the Introduce GUI to secure the service. Once these steps have been executed, Introduce will have generated the interfaces, common codes for compilation and deployment, and the service layout. At this point, the developer only needs to implement the backend processing logic for each method in the service. After completing the implementation of the backend logic, he/she can deploy the service using the Introduce GUI and it will automatically register itself in the Index Service.

4.5 Data Services Framework and Federated Query Support

The data services framework is a key component of the caGrid architecture in order to facilitate information sharing. Each caGrid 1.0 data service must conform to the standard query interface required by this framework and must expose standardized metadata describing its local domain model (i.e. the objects, and their relationships, to which it provides access). The application-specific implementation of the query interface must allow querying of any objects from the domain model and navigation of the model (i.e., the client should be able to traverse from an object to its associated objects via the associations described in the domain model). Introduce provides a plug-in extension for caGrid data services that creates a data service layout with the standard query interface; however, a developer can choose to implement the interface manually. The plug-in also generates a completely implemented data service for data sources developed using the caCORE SDK infrastructure, which requires no coding from the developer. Developers not using the caCORE SDK need only provide an implementation of the common query language against their local backend data store.

The basic query language of caGrid, which must be supported by every caGrid data service, is called caGrid Query Language (CQL). CQL is based on the notion of objects and object hierarchies. It allows the user to express queries that search for objects based on object classes and concept definitions and to specify criteria on the properties and associations of the objects. That is, it allows a client to define objects of interest by referencing their characteristics described in the logical domain model exposed by their containing service. A query expressed in CQL is defined in an XML document conforming to a well defined schema. The CQL schema includes 1) a *Target* element, which describes the data type that is being queried and that the query should return; 2) an *Attribute* element, which defines a predicate or restriction for an attribute of an object; 3) an *Association* element, which describes restrictions on an object via criteria over another associated object(specifically, it defines a relationship down the object model tree); and 4) a *Group* element, which defines logical joins of two or more conditions that operate against the object to which they are attached. Groups must have two or more children, which may be a mixture of Attribute, Association, or Group. The language is recursively designed, in that criteria can be defined in terms of restrictions on other criteria. CQL is designed to be sufficiently descriptive to meet the needs of data consumers, while being simple enough to facilitate integration of a variety of backend data resources.

The data services infrastructure supports a rich set of queries to allow researchers to obtain and integrate information from disparate data sources. The types of queries range from selection of data subsets from a single data source to joins and data aggregation across multiple data sources to queries on semantic meaning of and relationships between data objects representing biological entities. The following example illustrates a simple query expressed using CQL:

```
<CQLQuery xmlns="http://CQL.caBIG/1/gov.nih.nci.cagrid.CQLQuery">
        <Target name="gov.nih.nci.image.domain.CTChestImage">
                <Group logicRelation="AND">
                        <Attribute name="study" value="9011" predicate="EQUAL_TO"/>
                        <Attribute name="follow-up" value="yes" predicate="EQUAL_TO"/>
                </Group>
        </Target>
</CQLQuery>
```

This query searches for and retrieves images obtained in study 9011 which are longitudinal follow-up images. The researcher in our example scenario can submit this query to one or more caGrid data services as an XML document containing the query specification. Upon receiving the query, the service executes the query against its backend data store (possibly interpreting from the logical model to an internal model, such as a relational database) and generates a result set. The results are then serialized as XML by the data service infrastructure and returned to the client as a single query results set object. caGrid 1.0 APIs can iterate over the result set as either raw XML strings or de-serialized programming language objects.

The query process, especially execution of joins between multiple data sets, requires support for distributed execution since data sources can be located at disparate institutions. The

Federated Query Processing infrastructure (FQP) of caGrid 1.0 provides a mechanism to perform basic distributed aggregations and joins over multiple data services. An extension to CQL is supported to describe distributed query scenarios, such as the specification of multiple target data services, and the notion of distributed join criteria. The FQP contains three main client-facing components: an API implementing the business logic of federated query support, a Grid service providing remote access to that engine, and a Grid service for managing status and results of queries that were invoked asynchronously using the query service. The following example illustrates a simple federated query in our example scenario:

```
<DCQLQuery xmlns=”http://caGrid.caBIG/1.0/gov.nih.nci.cagrid.dcql”>

    <TargetObject name=”gov.nih.nci.image.domain.CTChestImage”>
        <Group logicRelation="AND">

              <ForeignAssociation
                  targetServiceURL=”https://host2:8443/wsrf/sercices/cagrid/StudyRegistry”>
                    <JoinCondition localAttributeName=”patientID”
                          foreignAttributeName=”patientNo” predicate=”EQUAL_TO”/>
                    <ForeignObject name=”gov.nih.nci.cabig.Participant”>
                       <Attribute name=”age” value=”18” predicate=”GREATER_THAN”/>
                    </ForeignObject>
             </ForeignAssociation>

              <Attribute name="follow-up" value="yes" predicate="EQUAL_TO"/>

        </Group>
    </TargetObject>

    <targetServiceURL>
          https://host1:8443/wsrf/services/cagrid/CTData
    </targetServiceURL>
</DCQLQuery>
```

In this example, the query defines a join on the “patientID” attribute of the object “Participant”, which stores the information about study participants, and the “patientNo” attribute of the object “CTChestImage”, which stores information about chest image data. The image data is hosted by the “CTData” service. The information on the study participants is maintained by the “StudyRegistry” service. The query also specifies that only those images, which are longitudinal follow-up images and have been obtained from patients who are older than 18 years old, should be returned.

4.6 Support for Large Dataset Transfer

The results of a query from a data service or output from an analytical service can be very large, ranging from hundreds of megabytes to hundreds of gigabytes in size. Several ways of accessing and transferring large volumes of data has been implemented in and integrated to the caGrid 1.0 infrastructure. The effort to standardize a “bulk data transfer” interface for large data has also been started, which is intended to provide uniform mechanism by which clients may access data sets from arbitrary services. This work currently supports access via WS-Enumeration, WS-Transfer, and GridFTP. The bulk data transfer effort was mainly motivated by the caBIG in-vivo imaging middleware effort, which builds on caGrid 1.0, and is being employed in that middleware system. The “bulk data transfer” interface is aimed to be a common mechanism which, if supported by large data providing services, allows a variety of mechanisms for data transfer for clients of varying capabilities. The simplest mechanism, WS-Transfer, is the baseline provided for interoperability, and defines a simple “get” operation which returns a simple XML encoding of the complete result set. The second mechanism is based on the WS-Enumeration standard and its implementation in the Globus Toolkit. WS-Enumeration allows a client to enumerate over results provided by a Web Service (much like a Grid-enabled *cursor*). This support is integrated into the caGrid Data Service tools, providing a mechanism for retrieving query results incrementally. The final mechanism is GridFTP, which provides fault-tolerant, high-performance binary file transfer mechanism in the Grid. In our example scenario, the result set of a query can contain a large number of high-resolution chest CT images. In that case, the data transfer between the client program and the data service can be carried out using WS-Enumeration, in which the client retrieves images in the result set in small groups (e.g., one by one), or using GridFTP to transfer images in bulk quickly from the server to the client host.

4.7 Support for Grid Workflows

The caBIG^{TM} environment is expected to provide an increasing number of analytical and data services developed and deployed by different institutions. Powerful applications can be created by composing workflows that access multiple services, thus harnessing data and analytical methods exposed as services more effectively. In such applications, information can be queried and extracted from one or more data sources and processed through a network of analytical services. caGrid 1.0 provides a workflow management service to support the orchestration of Grid services using the industry standard Business Process Execution Language (BPEL) and the execution and monitoring of BPEL-defined workflows in a secure Grid environment. BPEL is an XML language for describing business process behavior based on web/grid services. BPEL is layered on top of other Web technologies such as Web Services Description Language (WSDL), XML Schema, and XPath, which makes it a perfect candidate for use in caGrid. The BPEL notation includes flow control, variables, concurrent execution, input and output, transaction scoping/compensation, and error handling. A BPEL process describes a business process, which often invoke Web/Grid services to perform functional tasks.

Using the caGrid workflow management service, the researcher in our example application scenario can create a workflow to analyze chest CT image data. In this workflow, the chest CT image data is obtained by a query from one or more image services. These images are sent to one or more CAD analytical services for analysis. The results of analyses from these services are sent to another analytical service, which carries out a comparison of images analyzed by different CAD algorithms. The researcher can express this workflow in BPEL by identifying 1) the image data services and the query to obtain the images, 2) the CAD analytical services for image analysis and their input parameters, 3) the analytical service for comparison of analysis results, and 4) the destination of the comparison results (e.g., the client application). He/she also describes in BPEL the overall workflow by specifying where the output of a service is sent to (i.e., the output from data services should be sent to the CAD analytical services; the output from those services should be sent to the analytical service for comparison). The researcher can submit this workflow to the workflow management service for execution, query the status of the workflow (if the workflow is submitted for asynchronous execution), pause the execution of the workflow, resume the paused workflow, and terminate the workflow.

5 Status Report and Conclusions

caGrid version 1.0 was released to the caBIG community in December 2006. The core software and associated tools can be downloaded from the following URL: https://cabig.nci.nih.gov/workspaces/Architecture/caGrid. The distribution also provides a portal that can be used to discover, monitor, and visually display information about services running in the environment. A number of applications and services have already been developed by early adopters. Information about these applications and services is also available at the same URL.

Building on Grid technologies, caGrid implements several additional features to facilitate more efficient and secure sharing of information, tools, and applications. Key characteristics of caGrid are 1) support for syntactic and semantic interoperability among community-provided resources through use of rich service metadata, strongly typed services, common data elements, and controlled vocabularies; 2) focus on ease-of-use of the underlying technologies through high-level toolkits; 3) support for federated access to information resources; and 4) a comprehensive security infrastructure. caGrid 1.0 is a major milestone in the caBIG™ program towards achieving the program goals. While caGrid 1.0 is designed to address use cases in cancer research, the requirements associated with discovery, analysis and integration of large scale data, and coordinated studies are common in other biomedical fields. In this respect, caGrid 1.0 is the realization of a framework that can benefit the entire biomedical community.

Acknowledgements: The authors would like to acknowledge the contributions of the entire caGrid technical team (https://cabig.nci.nih.gov/workspaces/Architecture/caGrid), the members of the Architecture Workspace, the NCICB caCORE team, and the caBIG™ community. The authors also would like to thank Patrick McConnell for his contributions on requirements for caGrid and being a liaison between the caGrid developer team and the caBIG™ application community, Tony Pan, Ashish Sharma, Barla Cambazoglu, and Eliot Siegel for the imaging use case, Ken Buetow for his vision of a Cancer Grid and his tireless efforts in making caBIG a reality, and Tim Huang and Pearlly Yan for the Integrative Cancer Biology Program use case. This work was supported in part by the National Cancer Institute (NCI) caGrid Developer grant 79077CBS10, the State of Ohio Board of Regents BRTT Program grants ODOD AGMT TECH 04-049 and BRTT02-0003, from The Ohio State University Comprehensive Cancer Center-Arthur G. James Cancer Hospital and Richard J. Solove Research Institute, and by the U.S. Dept. of Energy under Contract DE-AC02-06CH11357.

6 References

1. Saltz J, Oster S, Hastings S, et al. caGrid: Design and Implementation of the Core Architecture of the Cancer Biomedical Informatics Grid. Bioinformatics. 2006;22(15):1910-6.
2. Foster I. Globus Toolkit Version 4: Software for Service-Oriented Systems. Journal of Computational Science and Technology. 2006;21(4):523-30.
3. Grethe JS, Baru C, Gupta A, et al. Biomedical Informatics Research Network: Building a National Collaboratory to Hasten the Derivation of New Understanding and Treatment of Disease. Stud Health Technol Inform. 2005;112:100-9.
4. Peltier ST, Ellisman MH. The Biomedical Informatics Research Network. In: Foster I, Kesselman C, editors. The Grid 2, Blueprint for a New Computing Infrastructure. 2nd ed: Morgan Kaufmann; 2003.
5. Foster I, Czajkowski K, Ferguson D, et al. Modeling and Managing State in Distributed Systems: The Role of OGSI and WSRF. Proceedings of IEEE. 2005;93(3):604-12.
6. Foster I, Kesselman C, editors. The Grid: Blueprint for a New Computing Infrastructure. San Francisco: Morgan Kaufmann; 1999.
7. Foster I, Kesselman C, Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of Supercomputer Applications. 2001;15(3):200-22.
8. Amendolia SR, Brady M, McClatchey R, Mulet-Parada M, Odeh M, Solomonides A. MammoGrid: Large-Scale Distributed Mammogram Analysis. Proceedings of the XV111th Medical Informatics Europe Conference (MIE'2003). St. Malo, France; 2003. p. 194-9.
9. Solomonides A, McClatchey R, Odeh M, et al. MammoGrid and eDiamond: Grid Applications in Mammogram Analysis. Proceedings of the IADIS International Conference: e-Society 2003. Lisbon, Portugal.; 2003. p. 1032-3.
10. Tweed T, Miguet S. Medical Image Database on the Grid: Strategies for Data Distribution. Proceedings of HealthGrid'03, Lyon, France. 2003:152 -62.
11. Duque H, Montagnat J, Pierson JM, Brunie L, Magnin IE. DM2: A Distributed Medical Data Manager for Grids. Proceedings of BioGrid'03, the 3rd International Symposium on Cluster Computing and the Grid (CCGrid 2003); 2003. p. 606 -11.
12. Rogulin D, Estrella F, Hauer T, McClatchey R, Amendolia SR, Solomonides A. A Grid Information Infrastructure for Medical Image Analysis. Proceedings of the Distributed Databases and Processing in Medical Image Computing Workshop (DiDaMIC-2004); 2004.
13. Stevens RD, Robinson A, Goble CA. myGrid: Personalised Bioinformatics on the Information Grid. Bioinformatics. 2003;19:302-4.

14. Erberich SG, Bhandekar M, Nelson MD, Chervenak A, Kesselman C. DICOM Grid Interface Service for Clinical and Research PACS: A Globus Toolkit Web Service for Medical Data Grids. International Journal of Computer Assistant Radiology and Surgery. 2006;1(87):100-2.
15. Stevens R, McEntire R, Goble CA, et al. myGrid and the drug discovery process. Drug Discovery Today: BIOSILICO. 2004 July;2(4):140-8.
16. Espert IB, Garcáa VH, Quilis JDS. An OGSA Middleware for Managing Medical Images using Ontologies. Journal of Clinical Monitoring and Computing. 2005;19(4):295-305.
17. Fukuzaki A, Nagashima T, Ide K, et al. Genome-Wide Functional Annotation Environment forin OBIGrid. Grid Computing in Life Science, First International Workshop on Life Science Grid, LSGRID 2004. Kanazawa, Japan; 2004. p. 82-91.
18. Shahab A, Chuon D, Suzumura T, et al. Grid Portal Interface for Interactive Use and Monitoring of High-Throughput Proteome Annotation. Grid Computing in Life Science, First International Workshop on Life Science Grid, LSGRID 2004. Kanazawa, Japan; 2004. p. 53-67.
19. Foster I, Kesselman C, Tsudik G, Tuecke S. A Security Architecture for Computational Grids. Proceedings of the 5th ACM Conference on Computer and Communications Security Conference; 1998. p. 83-92.
20. Lang B, Foster I, Siebenlist F, Ananthakrishnan R, Freeman T. A Multipolicy Authorization Framework for Grid Security. 5th IEEE International Symposium on Network Computing and Applications. Cambridge, MA; 2006. p. 269-72.
21. Hastings S, Oster S, Langella S, Ervin D, Kurc T, Saltz J. Introduce: An Open Source Toolkit for Rapid Development of Strongly Typed Grid Services. Journal of Grid Computing (in press, available online). 2007.
22. Business Process Execution Language for Web Services, Version 1.0. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/; 2002.
23. Alexander J, Box D, Cabrera LF, et al. Web Services Enumeration (WS-Enumeration). http://wwww3org/Submission/WS-Enumeration/; 2006.
24. Alexander J, Box D, Cabrera LF, et al. Web Services Transfer (WS-Transfer). http://wwww3org/Submission/WS-Transfer/; 2004.
25. Allcock B, Breshanan J, Kettimuthu R, et al. The Globus Striped GridFTP Framework and Server. Supercomputing 2005 (SC 2005); 2005.
26. Gurcan M, Pan T, Sharma A, et al. GridImage: A Novel Use of Grid Computing to Support Interactive Human and Computer-Assisted Detection Decision Support. Journal of Digital Imaging (in press, available online). 2007.
27. Pan T, Gurcan M, Langella S, et al. GridCAD: Grid-based Computer-aided Detection System. RadioGraphics. 2007;27(3):889-97.
28. Booth D, Haas H, McCabe F, et al. Web Services Architecture. http://wwww3org/TR/ws-arch/; 2003.
29. Hastings S, Langella S, Oster S, Saltz J. Distributed Data Management and Integration: The Mobius Project. Proceedings of GGF11 Semantic Grid Applications Workshop, Honolulu, Hawaii, USA. 2004:20-38.
30. Covitz PA, Hartel F, Schaefer C, et al. caCORE: A Common Infrastructure for Cancer Informatics. Bioinformatics. 2003;19(18):2404-12.
31. caBIG Compatibility Guidelines. https://cabig.nci.nih.gov/guidelines_documentation/caBIGCompatGuideRev2_final.pdf; 2005.
32. Basney J, Humphrey M, Welch V. The MyProxy Online Credential Repository. Software: Practice and Experience. 2005;35(9):801-16.
33. Bhatia K, Chandra S, Mueller K. GAMA: Grid Account Management Architecture. The First International Conference on e-Science and Grid Computing (e-Science '05). Melbourne, Australia; 2005.

34. Foster I, Kesselman C, Tuecke S, et al. A National Scale Authentication Infrastructure. IEEE Computer 2000;33(12):60-6.
35. Langella S, Oster S, Hastings S, Siebenlist F, Kurc T, Saltz J. Dorian: Grid Service Infrastructure for Identity Management and Federation. The 19th IEEE Symposium on Computer-Based Medical Systems. Salt Lake City, Utah; 2006.
36. Langella S, Oster S, Hastings S, Siebenlist F, Kurc T, Saltz J. Enabling the Provisioning and Management of a Federated Grid Trust Fabric. 6th Annual PKI R&D Workshop, Gaithersburg, MD. 2007 April.
37. OASIS Security Services (SAML) TC. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security; 2007.
38. Phillips J, Chilukuri R, Fragoso G, Warzel D, Covitz PA. The caCORE Software Development Kit: Streamlining construction of interoperable biomedical information services. BMC Medical Informatics and Decision Making. 2006;6(2).

Appendix

In this section we briefly review the Globus Toolkit, Mobius, and the caCORE systems that are leveraged by the caGrid infrastructure.

Globus Toolkit. The Globus toolkit(2) is the de facto standard implementation of the Web Services based specifications that make up WSRF. Globus development was initially focused at Argonne National Laboratory and the Information Sciences Institute at University of Southern California, but now includes a large international community. The project provides middleware and tools, which extend the Axis toolkit to enable creation of grid services; many standard services useful in Grid environments, such as indexing, delegation, monitoring, security, credential management, naming, job management, and data transfer; and a runtime infrastructure.

Mobius. The Mobius framework(29) contains several services that in combination enable the creation of a strongly typed data integration environment. A key component of Mobius that is leveraged in caGrid is the GME (Global Model Exchange) service. In order for services to communicate in the Grid, data exchanged between services must be described in a format that is understood by all components involved. This requires a mechanism for defining metadata and data via a consistent modeling pattern, distinct instances of which we call data models. A data model is the specification of the structure, format, syntax, and occurrences of the data instances it represents. The GME provides support for storing, versioning, and discovering data models, which can be used as input or output data types by Grid service operations.

caCORE. The NCI Common Ontologic Reference Environment (caCORE)(30) includes the Enterprise Vocabulary Service (EVS), the Cancer Data Standards Repository (caDSR) and the Software Development Kit (SDK). The EVS provides the description logic concepts and terms used to describe the classes, attributes, and instance data in caGrid. The caDSR is an ISO/IEC 11179 metadata registry that provides the authoritative registry of caGrid data elements and their corresponding semantics. The caCORE SDK provides a convenient mechanism for creating model driven, semantically interoperable data resources that can then be accessed by caGrid(38).